Quantum algorithms to check Resiliency, Symmetry and Linearity of a Boolean function

Kaushik Chakraborty · Anupam Chattopadhyay · Subhamoy Maitra

Abstract In this paper, we present related quantum algorithms to check the order of resiliency, symmetry and linearity of a Boolean function that is available as a black-box (oracle). First we consider resiliency and show that the Deutsch-Jozsa algorithm can be immediately used for this purpose. We also point out how the quadratic improvement in query complexity can be obtained over the Deutsch-Jozsa algorithm for this purpose using the Grover's technique. While the worst case quantum query complexity to check the resiliency order is exponential in the number of input variables of the Boolean function, we require polynomially many measurements only. We also describe a subset of n-variable Boolean functions for which the algorithm works in polynomially many steps, i.e., we can achieve an exponential speed-up over best known classical algorithms. A similar kind of approach can be exploited to check whether a Boolean function is symmetric (respectively linear) or not. Given a Boolean function as an oracle, it is important to devise certain algorithms to test whether it has a specific property or it is ϵ-far from having that property. The efficiency of the algorithm is judged by the number of calls to the oracle so that one can decide, with high probability, between these two alternatives. We show that this can be achieved in $O(\epsilon^{-\frac{1}{2}})$ query complexity. This is obtained by showing that the problem of checking symmetry or linearity can be efficiently reduced to testing whether a Boolean function is constant.

This is a thoroughly revised and extended version of the paper "Quantum algorithms to check Resiliency of a Boolean function" by Kaushik Chakraborty and Subhamoy Maitra that has been presented in WCC 2013, April 15-19, 2013, Bergen, Norway. Sections 3, 4 contain additional materials over the workshop version.

Kaushik Chakraborty
Indian Statistical Institute, Kolkata 700 108, India
E-mail: kaushik.chakraborty9@gmail.com

Anupam Chattopadhyay
UMIC Research Centre, RWTH Aachen University, Aachen, Germany
E-mail: anupam@umic.rwth-aachen.de

Subhamoy Maitra
Applied Statistics Unit, Indian Statistical Institute, Kolkata 700 108, India
E-mail: subho@isical.ac.in

Keywords Boolean Functions, Deutsch-Jozsa Algorithm, Grover's Algorithm, Linearity, Measurement, Property Testing, Resiliency, Symmetry.

Mathematics Subject Classification (2000) 68Q12

1 Introduction

After the introduction of the Deutsch-Jozsa algorithm [7], several works have been presented in the literature to describe strategies that can distinguish Boolean functions of different weights (for example, see [5] and the references therein). Such problems are actually related to studying the Walsh spectrum of Boolean functions. From a cryptologic viewpoint, the concept of balancedness can be generalized to resiliency and we will consider this problem first. We will study how efficiently, in quantum paradigm, one can check whether a Boolean function is m-resilient or not. In a similar direction we also analyse the problem of checking symmetry and linearity of a Boolean function.

1.1 Basics of Boolean functions

A Boolean function on n variables may be viewed as a mapping from $\{0,1\}^n$ into $\{0,1\}$. We will denote the set of n-variable Boolean functions as $\mathcal{B}_n$. It is easy to note that $|\mathcal{B}_n| = 2^{2^n}$.

An n-variable Boolean function $f(x_1,\ldots,x_n)$ can be considered to be a multivariate polynomial over $GF(2)$. This polynomial can be expressed as a $GF(2)$ sum of all distinct k-th order product terms $(0 \le k \le n)$ of the variables. More precisely, $f(x_1,\ldots,x_n)$ can be written as

$$a_0 \oplus \bigoplus_{1\le i\le n} a_i x_i \oplus \bigoplus_{1\le i<j\le n} a_{ij} x_i x_j \oplus \ldots \oplus a_{12\ldots n} x_1 x_2 \ldots x_n,$$

where the coefficients $a_0, a_{ij}, \ldots, a_{12\ldots n} \in \{0,1\}$. This representation of f is called the algebraic normal form (ANF) of f. The number of variables in the highest order product term with nonzero coefficient is called the algebraic degree, or simply the degree of f and denoted by $deg(f)$.

Functions of degree at most one are called *affine* functions. An affine function with constant term equal to zero is called a *linear* function. The set of all n-variable affine functions is denoted by $\mathcal{A}_n$. That is the set of affine functions contains all the linear functions and their complements. By abuse of notation, we will use the terms "linear" and "affine" interchangeably throughout this document. For several of our reductions, we will use the constant functions (all zero and all one). Let us denote the set of n-variable constant functions by $\mathcal{C}_n$.

Let $x = (x_1,\ldots,x_n)$ and $\omega = (\omega_1,\ldots,\omega_n)$ both belong to $\{0,1\}^n$ and the inner product

$$x \cdot \omega = x_1\omega_1 \oplus \cdots \oplus x_n\omega_n.$$

A Boolean function $l(x)$ is linear if it can be written as $l(x) = \omega \cdot x$ for some fixed ω. However, by extending the notation, we also consider the affine functions of the form $l(x) = \omega_0 \oplus \omega \cdot x$, where $\omega_0 \in \{0,1\}$ as linear functions.

An n-variable Boolean function f is called symmetric if

$$f(x) = f(y) \text{ for all } x, y \in \{0,1\}^n \text{ such that } wt(x) = wt(y).$$

We denote the set of n-variable symmetric Boolean functions as $\mathcal{SB}_n$.

Let $f(x)$ be a Boolean function on n variables. Then the *Walsh transform* of $f(x)$ is an integer valued function over $\{0,1\}^n$ which is defined as

$$W_f(\omega) = \sum_{x\in\{0,1\}^n} (-1)^{f(x)\oplus x\cdot\omega}.$$

The fastest known classical algorithm to calculate all the Walsh spectrum values of $f \in \mathcal{B}_n$, i.e., $W_f(\omega)$ at each of the 2^n points ω, is of $O(n2^n)$ time complexity. Calculation of the Walsh spectrum value at a specific point requires $O(2^n)$ time in classical domain.

The non-linearity (or non-affinity) of an n-variable function f is

$$nl(f) = min_{g\in\mathcal{A}_n}(d(f,g)),$$

i.e., the distance from the set of all n-variable affine functions. In terms of Walsh spectrum, the non-linearity of f is given by

$$nl(f) = 2^{n-1} - \frac{1}{2}\max_{\omega\in\{0,1\}^n} |W_f(\omega)|.$$

For a binary string *str*, the number of 1's in the string is called the (Hamming) weight of *str* and denoted as $wt(str)$. In truth table representation, a Boolean function $f \in \mathcal{B}_n$ can be viewed as a binary string of length 2^n, which is the output column of the truth table. As we have already discussed, if $wt(f) = 0$ or 2^n, then it is called a constant Boolean function. If $wt(f) = 2^{n-1}$, then f is called a balanced function. In terms of the Walsh spectrum, $f \in \mathcal{B}_n$ is balanced if and only if $W_f(0,0,\ldots,0) = 0$. Following [9], a function $f \in \mathcal{B}_n$ is m-resilient if and only if its Walsh transform satisfies

$$W_f(\omega) = 0, \text{ for } 0 \le wt(\omega) \le m.$$

It is easy to note that a balanced function is actually a 0-resilient function. Thus, informally speaking, the problems related to resiliency are a generalization of the problems related to balancedness.

1.2 The algorithmic issues

Let us now briefly discuss certain algorithms in classical as well as quantum domain. We consider that the Boolean function $f \in \mathcal{B}_n$ is available as an (classical or its quantum counterpart) oracle, i.e., the corresponding output can be obtained efficiently (in constant time) given the input. This is termed as a "query". Given the promise that f is either constant or balanced, to check which one it is, we have the Deutsch-Jozsa [7] algorithm to solve it in a single query. Though there is no deterministic polynomial time algorithm to solve this problem in the classical domain, probabilistic polynomial time algorithms are indeed available to solve this problem efficiently. Given that $W_f(\omega) = 0$ or $\pm 2^n$, the question of "which one it is" can be solved exactly in a similar manner, by considering the function $f(x) \oplus \omega \cdot x$ instead of $f(x)$.

It is well known that checking resiliency of an n-variable Boolean function requires exponential number of queries in n in the classical domain. In this paper we try to analyse the solution of this problem in the quantum domain. We note that the Deutsch-Jozsa [7] algorithm can be used for this purpose. Further, we try to devise strategies with better efficiency than this using the Grover's algorithm [8]. The Grover's algorithm has earlier been used in weight decision problems for Boolean functions [5] and we note that a more involved application of this algorithm can also be exploited in the resiliency checking problem. In our proposed quantum algorithms[1], though the worst case query complexity[2] is exponential, we need only polynomially many measurements in the computational basis[3] for this purpose. We also identify a sub-class of Boolean functions for which our quantum algorithms work with polynomially many queries in n. The best known classical algorithm for this sub-class requires exponentially many steps.

1.3 Property testing

We also interpret the results in terms of property testing of a Boolean function. Consider that a Boolean function is implemented inside a black box and one can obtain the output given an input in constant time. Each such operation may be referred to as a query. One may like to test several properties of the Boolean function by exploiting such queries. Naturally, it is important to test the properties with as few queries as possible and with a good probability of success. Testing whether a Boolean function (given as an oracle) is linear or not is an important question in the field of computational complexity [3,2]. For further results in this area of property testing, one may refer to [1,10,11].

Definition 1 Given two n-variable Boolean functions f and g, we define f, g as ϵ-far if

$$\frac{|\{x \in \{0,1\}^n : f(x) \neq g(x)\}|}{2^n} = \frac{d(f,g)}{2^n} \geq \epsilon,$$

i.e., if the Hamming distance $d(f,g)$ between the truth tables of f and g is at least $\epsilon 2^n$. Further, an n-variable Boolean function f will be called ϵ-far from a subset S of n-variable Boolean functions if f is ϵ-far from all the functions $g \in S$.

The first property we are interested here is checking whether a Boolean function is m-resilient or is ϵ-far $(0 < \epsilon \leq 1)$ from having that property. However, since the set of resilient Boolean functions are yet to be characterized and the definition is related to Walsh spectrum rather than the truth table, it is not immediate how to map it to the framework of property testing. On the other hand, for the other properties, such as symmetry and linearity, the definition in the domain of property testing works perfectly and we present improved results over those given in [10]. The quantum algorithm [10] for testing whether a function is symmetric

[1] Our analysis here does not require any specialized knowledge of the quantum paradigm. Instead the work is mostly related to combinatorial properties of Boolean functions. We show how such properties of Boolean functions can be exploited to achieve novel and improved results in the field of quantum algorithms.

[2] For quantum algorithms, we write "query complexity" instead of "time complexity" as we need to query some oracles, e.g., $U_f, \mathcal{O}_g$ as described in Section 2.

[3] For more details on query complexity and measurements, refer to [15].

(respectively linear) or ϵ-far ($0 < \epsilon < \frac{1}{2}$) from symmetric (respectively linear) functions requires $O(\epsilon^{-\frac{2}{3}})$ many calls, whereas our technique requires only $O(\epsilon^{-\frac{1}{2}})$ many queries.

The organization of the paper is as follows. In Section 2, we present the quantum algorithm for resiliency checking and its analysis in detail. Next, in Section 3, we discuss how a similar idea can be exploited to devise a quantum algorithm to test whether a function is symmetric. We further use similar ideas in Section 4 towards linearity testing. Section 5 concludes the paper.

2 Algorithm to check Resiliency

Given f either constant or balanced, if the corresponding quantum implementation U_f is available, Deutsch-Jozsa [7] provided a quantum algorithm that decides in a constant number of queries which one it is. The overall idea of the algorithm can be summarized by Figure 1 and the step by step description of the Deutsch-Jozsa [7] algorithm can be written as in Algorithm 1.

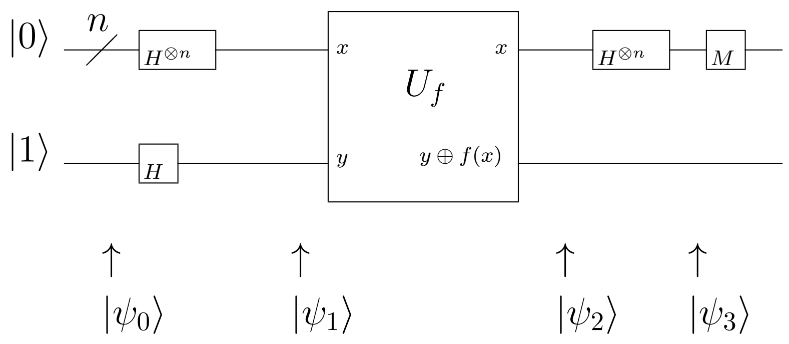

Fig. 1 Quantum circuit to implement the Deutsch-Jozsa Algorithm

Let us now describe our interpretation of the Deutsch-Jozsa algorithm in terms of Walsh spectrum values. We denote the operator for Deutsch-Jozsa algorithm as $\mathcal{D}_f = H^{\otimes n} U_f H^{\otimes n}$, where the Boolean function f is available as an oracle U_f. For brevity, we abuse the notation and do not write the auxiliary qubit, i.e., $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ and the corresponding output in this case[4]. Now one can observe that $\mathcal{D}_f |0\rangle^{\otimes n} = \sum_{z \in \{0,1\}^n} \sum_{x \in \{0,1\}^n} \frac{(-1)^{x \cdot z \oplus f(x)}}{2^n} |z\rangle = \sum_{z \in \{0,1\}^n} \frac{W_f(z)}{2^n} |z\rangle$, i.e., the associated probability for the state $|z\rangle$ is $\frac{W_f^2(z)}{2^{2n}}$. In this regard, we have the following technical result as pointed out in [13].

Proposition 1 *Given $f \in \mathcal{B}_n$, $\mathcal{D}_f |0\rangle^{\otimes n}$ produces a superposition of all states $z \in \{0,1\}^n$ with the amplitude $\frac{W_f(z)}{2^n}$ corresponding to each state $|z\rangle$.*

[4] We go for similar abuse of notation for the phase inversion oracle later.

```
    Input: A Boolean function $f \in \mathcal{B}_n$, available in the form of the unitary
           transformation $U_f$
    Output: $n$-bit pattern
1   Take an $(n+1)$ qubit state $|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle$;
2   Apply Hadamard Transform $H^{\otimes(n+1)}$ on $|\psi_0\rangle$ to get
        $|\psi_1\rangle = \sum_{x\in\{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$;
3   Apply $U_f$ on $|\psi_1\rangle$ to get
        $|\psi_2\rangle = \sum_{x\in\{0,1\}^n} \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$;
4   Apply Hadamard Transform on the first $n$ qubits of $|\psi_2\rangle$ to obtain
        $|\psi_3\rangle = \sum_{z\in\{0,1\}^n} \sum_{x\in\{0,1\}^n} \frac{(-1)^{x\cdot z\oplus f(x)}|z\rangle}{2^n} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$;
5   Measurement at $M$: measure the first $n$ qubits of $|\psi_3\rangle$;
6   After measurement, the all zero state ($n$-bit all zero pattern) implies
    that the function is constant, else it is balanced;
```

Algorithm 1: The Deutsch-Jozsa algorithm [7].

Consider that we are interested to know whether $f \in \mathcal{B}_n$ is m-resilient. Let $S_m = \{x \in \{0,1\}^n | wt(x) \leq m\}$ and $\overline{S}_m = \{x \in \{0,1\}^n | wt(x) > m\}$. Consider the n-qubit state $|\Psi\rangle = \sum_{s\in S_m} \frac{W_f(s)}{2^n}|s\rangle + \sum_{s\in \overline{S}_m} \frac{W_f(s)}{2^n}|s\rangle$. For brevity, let us represent this as $|\Psi\rangle = a|X\rangle + b|Y\rangle$, where $|X\rangle = \sum_{s\in S_m} \frac{W_f(s)}{a2^n}|s\rangle$, $|Y\rangle = \sum_{s\in \overline{S}_m} \frac{W_f(s)}{b2^n}|s\rangle$, $a^2 = \sum\limits_{s\in S_m} \frac{W_f^2(s)}{2^{2n}}$ and $b^2 = \sum\limits_{s\in \overline{S}_m} \frac{W_f^2(s)}{2^{2n}}$.

Using the Deutsch-Jozsa algorithm, we obtain $\sum_{z\in\{0,1\}^n} \frac{W_f(z)}{2^n}|z\rangle$ (before the measurement). That is, some state $s \in S_m$ will appear after the measurement with probability a^2. Hence, after $O(\frac{1}{a^2})$ iterations, one can expect to observe some $s \in S_m$ after the measurement and output that f is not m-resilient. If f is indeed m-resilient, then $a^2 = 0$ and thus any state $s \in S_m$ will never appear at the output. One may note that the minimum absolute value of the Walsh spectrum is 2. Thus, we can have a situation that f is not m-resilient, but a^2 is $O(\frac{1}{2^{2n}})$. In such a case, the algorithm will require exponentially many queries to provide the correct result. Thus the resiliency checking algorithm is as in Algorithm 2.

Theorem 1 *Algorithm 2 correctly answers NO, but answers YES with constant success probability in* $O(\frac{1}{a^2})$ *queries, where* $a^2 = \sum\limits_{s\in S_m} \frac{W_f^2(s)}{2^{2n}} > 0$.

Proof According to Algorithm 2, one can observe that for each iteration, the success probability is a^2. After i many queries, the success probability will be $1-(1-a^2)^i$. When i is $O(\frac{1}{a^2})$, there exists a positive constant $c < 1$ such that $1-(1-a^2)^i > c$. □

```
    Input: A function $f \in \mathcal{B}_n$, available in the form of the unitary transformation $U_f$,
           order of resiliency $m$ and the number of iterations $r$
    Output: YES/NO
1   $S_m = \{x \in \{0,1\}^n | wt(x) \le m\}$;
2   for $r$ many times do
3       Apply Deutsch-Jozsa algorithm and take the $n$-bit output $u$;
4       if $u \in S_m$ then
            Report that the function is not $m$-resilient (NO) and terminate;
        end
    end
5   Report that the function is $m$-resilient (YES);
```

Algorithm 2: Resiliency checking using the Deutsch-Jozsa algorithm [7].

Remark 1 Algorithm 2 is written in such a manner that if a function is indeed m-resilient, then $a = 0$ and thus the algorithm will say YES after executing r steps. However, it is known that for nonzero Walsh spectrum values, the minimum is ± 2 and thus, $a^2 \ge \frac{4}{2^{2n}}$. Hence, after repeating the algorithm r, i.e., $O(2^{2n})$ times, if we don't observe any binary string $u \in S_m$ after measurements, then we can conclude that the Boolean function f is m-resilient with success probability greater than some predefined constant c. This provides the worst case scenario.

2.1 Improvement using the Grover's Algorithm

Grover's algorithm [8] provides a quadratic speed-up compared to repeated use of the Deutsch-Jozsa algorithm and that is what we try out here. Instead of equal superposition $|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{2^{\frac{n}{2}}}\sum_{x \in \{0,1\}^n} |x\rangle$ in the Grover's algorithm, we will use a state of the form $|\Psi\rangle = \mathcal{D}_f(|0\rangle^{\otimes n}) = \sum_{x \in \{0,1\}^n} \frac{W_f(x)}{2^n}|x\rangle$.

Consider that any n-qubit state is represented in the computational basis. We want to amplify the amplitude of the points in S_m. This we achieve in a similar manner as in Grover's algorithm.

Grover's algorithm requires inversion of phase. Towards this, we will use $g(x) \in \mathcal{B}_n$, different from $f(x)$. The corresponding operator $\mathcal{O}_g$ inverts the phase of the states $|x\rangle$ where $x \in S_m$. That is, we need to change phase for the points having weight less than or equal to m. This can be achieved by choosing the n-variable Boolean function $g(x)$ such that $g(x) = 1$, when $wt(x) \le m$, and $g(x) = 0$, otherwise. Thus g is a symmetric function. A symmetric Boolean function can be efficiently implemented, as described in [6]. The circuit complexity of an n-variable symmetric Boolean function is $4.5n + o(n)$. It is known that given a classical circuit g, a quantum circuit of comparable efficiency can be implemented, with some very small number of extra garbage bits. Thus, we will consider that for a symmetric function g, the quantum circuit $\mathcal{O}_g$ can be efficiently implemented using $O(n)$ circuit complexity.

Now let us consider the operator $G_t = [(2|\Psi\rangle\langle\Psi| - I)\mathcal{O}_g]^t$ acting on $|\Psi\rangle$ to get $|\Psi_t\rangle$. The idea presented in the following result is easy to see as it follows the amplitude amplification used in the Grover's algorithm. However, we present the proof for better understanding.

Proposition 2 *Let* $|\Psi\rangle = \sum_{s\in S_m} \frac{W_f(s)}{2^n}|s\rangle + \sum_{s\in \overline{S}_m} \frac{W_f(s)}{2^n}|s\rangle = a|X\rangle + b|Y\rangle$*, where* $a = \sin\theta$*,* $b = \cos\theta$*,* $|X\rangle = \sum_{s\in S_m} \frac{W_f(s)}{a2^n}|s\rangle$*,* $|Y\rangle = \sum_{s\in \overline{S}_m} \frac{W_f(s)}{b2^n}|s\rangle$*. The application of the* $[(2|\Psi\rangle\langle\Psi| - I)\mathcal{O}_g]^t$ *operator on* $|\Psi\rangle$ *produces* $|\Psi_t\rangle$*, in which the amplitude of* $|X\rangle$ *is* $\sin(2t+1)\theta$*.*

Proof For $t = 1$, one can check that $|\Psi_1\rangle = [(2|\Psi\rangle\langle\Psi| - I)\mathcal{O}_g]|\Psi\rangle = [(2|\Psi\rangle\langle\Psi|)\mathcal{O}_g]|\Psi\rangle - \mathcal{O}_g|\Psi\rangle$. Now substituting the values of a, b we get that $|\Psi_1\rangle = \sin 3\theta|X\rangle + \cos 3\theta|Y\rangle$.

Now we will use induction. Let the application of $[(2|\Psi\rangle\langle\Psi| - I)\mathcal{O}_g]^t$ operator on $|\Psi\rangle$ update the amplitude of $|X\rangle$ as $\sin(2t\theta + \theta)$, for $t = k$. From the assumption we have $[(2|\Psi\rangle\langle\Psi| - I)\mathcal{O}_g]^k|\Psi\rangle = \sin(\theta + 2k\theta)|X\rangle + \cos(\theta + 2k\theta)|Y\rangle$. Now for $t = k+1$, it can be checked that $[(2|\Psi\rangle\langle\Psi| - I)\mathcal{O}_g]^{(k+1)}|\Psi\rangle = \sin(\theta + 2(k+1)\theta)|X\rangle + \cos(\theta + 2(k+1)\theta)|Y\rangle$. Thus, the proof. □

```
    Input: A Boolean function $f \in \mathcal{B}_n$, available in the form of the unitary transformation
           $U_f$, order of resiliency $m$ and the number of iterations $r$ and a series of
           positive integers $t_i$, $1 \le i \le r$ related to the number of the Grover's iterations
    Output: YES/NO
1   $S_m = \{x \in \{0,1\}^n | wt(x) \le m\}$;
2   for $i = 1$ to $r$ do
3       Apply Deutsch-Jozsa algorithm till the step before measurement to obtain
        $|\Psi\rangle = \sum_{s\in S_m} \frac{W_f(s)}{2^n}|s\rangle + \sum_{s\in \overline{S}_m} \frac{W_f(s)}{2^n}|s\rangle$;
4       By applying the Grover's iteration, obtain $|\Psi_{t_i}\rangle = [(2|\Psi\rangle\langle\Psi| - I)\mathcal{O}_g]^{t_i}|\Psi\rangle$;
5       Measure $|\Psi_{t_i}\rangle$ in the computational basis to obtain $n$-bit string $u$;
6       if $u \in S_m$ then
            Report that the function is not $m$-resilient (NO) and terminate;
        end
    end
7   Report that the function is $m$-resilient (YES);
```

Algorithm 3: Resiliency checking using the Grover's algorithm [7].

After the Deutsch-Jozsa algorithm we obtain $\sum_{z\in\{0,1\}^n} \frac{W_f(z)}{2^n}|z\rangle$ (before the measurement) with $a^2 = \sum\limits_{s\in S_m} \frac{W_f^2(s)}{2^{2n}}$ and $b^2 = \sum\limits_{s\in \overline{S}_m} \frac{W_f^2(s)}{2^{2n}}$. Thus, we have $\sin\theta = a$. For large n, one can approximate it as $\theta = a$ and hence we need t iterations of a Grover like strategy such that $(2t+1)\theta \ge \sin^{-1} c$, where c is a predefined constant. Thus, here we need an expected $O(\frac{1}{a})$ iterations, compared to $O(\frac{1}{a^2})$ iterations using the Deutsch-Jozsa algorithm only. One important issue here is that any estimate of a may not be known and thus, estimating t_r could be challenging. Given that t_r can be estimated, after application of the Grover's algorithm, we will obtain a state $\sum\limits_{s\in S_m} a'_s|s\rangle + \sum\limits_{s\in \overline{S}_m} b'_s|s\rangle = a'|X\rangle + b'|Y\rangle$, where $(a')^2$ is very close to 1. Using this (Grover's algorithm followed by Deutsch-Jozsa algorithm), we get a quadratic speed-up over just using Deutsch-Jozsa algorithm.

It is natural to use the Grover's algorithm for amplitude amplification and thus obtaining quadratic speed-up. However, in the known applications (e.g., search), the number of target states, for which the amplitude is increased, is not large. That guarantees the efficient implementation of the phase reversal circuit. In our case, the situation is different as we need to amplify the amplitude at $\sum_{i=0}^{m}\binom{n}{i}$ points of weight $\leq m$ and this could be exponential. Thus, it is an important question whether the phase reversal can be implemented efficiently. In our case, this can be achieved as the phase reversal can be implemented with symmetric functions, the implementation of which is efficient [6].

2.2 Deciding the number of Grover's iteration

One may refer to [4] to study the detailed analysis related to the number of Grover's iterations required to obtain a success probability close to 1. However, we like to present our analysis in detail for a better understanding of how many iterations we need and at the same time the number of measurements needed. Thus, let us explicitly describe how one can decide the values of t_i for $1 \leq i \leq r$.

As given in Algorithm 3, we have $|\Psi\rangle = \sum_{s\in S_m} \frac{W_f(s)}{2^n}|s\rangle + \sum_{s\in \overline{S}_m} \frac{W_f(s)}{2^n}|s\rangle = a|X\rangle + b|Y\rangle$, where $a = \sin\theta$, $b = \cos\theta$. Our motivation is to observe some state $s \in S_m$, if $\sum_{s\in S_m} \frac{W_f(s)}{2^n} > 0$, i.e., if $a > 0$. We will apply Grover's algorithm to obtain $|\Psi_{t_i}\rangle = [(2|\Psi\rangle\langle\Psi| - I)\mathcal{O}_g]^{t_i}|\Psi\rangle = \sin\theta_i|X\rangle + \cos\theta_i|Y\rangle$ such that $\sin\theta_i$ is greater than or equal to some predefined constant, say $\sin\theta_c = c$. Thus, we require that there should exist some t_i such that $\theta_c \leq (2t_i+1)\theta \leq \pi - \theta_c$.

We start with $t_1 = 0$, i.e., we expect that $\theta_c \leq (2t_1+1)\theta = \theta \leq \pi - \theta_c$. If this is the case, we are done. That is, in this case, we do not apply the Grover's algorithm at all and the situation is similar to Algorithm 2 where only the Deutsch-Jozsa algorithm is used.

We divide the angular region $[0, \frac{\pi}{2}]$ in r angular grids,

$$\{[\alpha_{r+1}, \alpha_r], [\alpha_r, \alpha_{r-1}], \ldots, [\alpha_2, \alpha_1]\},$$

where $0 = \alpha_{r+1} < \alpha_r < \ldots < \alpha_2 < \alpha_1 = \frac{\pi}{2}$. We will now show how to select the values of $\alpha_r, \ldots, \alpha_2$, other than the boundary values α_{r+1}, α_1. There must exist some $i \in [1, r]$ such that $\alpha_{i+1} \leq \theta < \alpha_i$. For that, we need the t_i value such that $\theta_c = (2t_i+1)\alpha_{i+1} \leq (2t_i+1)\theta < (2t_i+1)\alpha_i = \pi - \theta_c$. Thus we must satisfy $\theta_c = (2t_i+1)\alpha_{i+1} < (2t_i+1)\alpha_i = \pi - \theta_c$ for each i. For each of them, we need to satisfy the following.

$$(2t_i+1)\alpha_i = \pi - \theta_c, \tag{1}$$

$$(2t_i+1)\alpha_{i+1} = \theta_c. \tag{2}$$

Similarly, we have

$$(2t_{i-1}+1)\alpha_{i-1} = \pi - \theta_c, \tag{3}$$

$$(2t_{i-1}+1)\alpha_i = \theta_c. \tag{4}$$

Thus, from (1), (4), we get,

$$\frac{2t_i+1}{2t_{i-1}+1} = \frac{\pi - \theta_c}{\theta_c}. \tag{5}$$

Taking the initial condition $t_1 = 0$ and by solving the above recurrence relation, we get,

$$(2t_i+1) = \left(\frac{\pi - \theta_c}{\theta_c}\right)^{i-1} \tag{6}$$

This provides us the values of t_i's and that in turn will decide how the angular grids will be chosen. Now we need to decide the value of r and for that we have to consider the worst case given the value of $\sin\theta$. Let $\sin\theta = a$. From Proposition 2 we know that to ensure $\sin\theta$ is close to 1, the maximum value among the t_i's, i.e., t_r according to our technique, should be taken as $O(\frac{1}{a})$. So, $(2t_r + 1) \approx \frac{1}{a}$ and we can write $r \approx \log_{\frac{\pi-\theta_c}{\theta_c}}(\frac{1}{a})$, i.e., r is $O(\log\frac{1}{a})$. Thus, we have the following result.

Theorem 2 *Algorithm 3 correctly answers NO, but answers YES with a constant success probability in $O(\log\frac{1}{a})$ measurements (each measurement is as in step 5) and the total number of execution of the Grover's operator (as in step 4) is $O(\frac{1}{a})$ where $a^2 = \sum_{s\in S_m} \frac{W_f^2(s)}{2^{2n}}$.*

Proof How we estimate r is explained above. In Algorithm 3, at the i-th step we apply the operator $[(2|\psi\rangle\langle\psi| - I)\mathcal{O}_g]$, t_i times. Here i varies from 1 to r. So, the total number of times the Grover's operator is applied is $T = \sum_{i=1}^{r} t_i$. From (6), we can substitute the value of t_i and get

$$T = \frac{1}{2}\sum_{i=1}^{r}\left(\left(\frac{\pi-\theta_c}{\theta_c}\right)^{i-1} - 1\right).$$

By solving this equation and also substituting the value of r, we get

$$T \approx \frac{1}{2}\left[\frac{1/a - 1}{(\pi-\theta_c)/\theta_c - 1} - \frac{1}{2}\left\{\log_{\frac{\pi-\theta_c}{\theta_c}}(\frac{1}{a})(\log_{\frac{\pi-\theta_c}{\theta_c}}(\frac{1}{a}) + 1)\right\}\right]. \tag{7}$$

So, the number of times the Grover's operator is executed is $O(\frac{1}{a})$. □

Remark 2 Similar to Remark 1, for Algorithm 3 we need to consider the case when the function is m-resilient, i.e., $a = 0$. In this case r will be $O(\log 2^n)$, i.e., $O(n)$ and t_r will be $O(2^n)$, that provides the worst case scenario.

Remark 3 We like to point out that the number of measurements in both Algorithm 2 and Algorithm 3 are r. In case of Algorithm 2, r is $O(\frac{1}{a^2})$ and can be exponential in n worst case. However, for Algorithm 3, r is $O(\log\frac{1}{a})$, which is polynomial in n in worst case. For Algorithm 2, the number of queries using the Deutsch-Jozsa operator is $r = O(\frac{1}{a^2})$ and for Algorithm 3, the number of queries using the Grover's operator is $T = O(\frac{1}{a})$ and both of them could be exponential in the worst case. In summary,

- in terms of number of queries, Algorithm 3 provides quadratic improvement over Algorithm 2, though both can be exponential in worst case;
- in terms of number of measurements, Algorithm 3 requires polynomially many measurements in worst case, while Algorithm 2 requires exponentially many.

Let us now study the algorithm from two aspects. First we see some special classes of Boolean functions for which the quantum algorithm requires query complexity polynomial in the number of input variables. Next, we interpret our results from a property testing point of view.

2.3 Checking m-resiliency among functions with three valued Walsh spectrum

From the analysis in the previous section, we note that the Deutsch-Jozsa algorithm or the Deutsch-Jozsa algorithm (without measurement) followed by the Grover's algorithm can be used to check whether a Boolean function is m-resilient or not. It is very clear that the second strategy provides a quadratic speed-up over the first one. It is also evident that the quantum algorithms, in worst case, may take exponentially many queries in n. Thus, it would be interesting to consider a class of Boolean functions for which the problem can be solved in polynomially many queries in n in the quantum paradigm.

In [16–18], several characterizations and constructions of resilient functions have been presented. In particular, it has been pointed out in [16] that the Walsh spectrum values of any m-resilient function will be divisible by 2^{m+2}. In this direction, we will concentrate on Boolean functions with Walsh spectrum values that are multiples of 2^{m+2}. Let us define

$$\mathcal{A}_n = \{f \in \mathcal{B}_n | W_f(\omega) \equiv 0 \bmod 2^{m+2}\}.$$

In this case, if the function is not m-resilient, then $a \geq \frac{2^{m+2}}{2^n}$ and thus, the query complexity of checking resiliency is $O(2^{n-m-2})$ using Algorithm 3. In case, $m \geq n - O(poly(\log n))$, it is clear that the query complexity of checking resiliency is $O(poly(n))$.

We do not know of any classical algorithm that can efficiently decide whether a function $f \in \mathcal{A}_n$ is m-resilient. Thus, we get an exponential speed-up in this case using a quantum algorithm over classical ones.

2.4 Property testing interpretation

As discussed earlier, given two n-variable Boolean functions f and g, the existing literature [1,10,11] define f, g as ϵ-far if

$$\frac{|\{x \in \{0,1\}^n : f(x) \neq g(x)\}|}{2^n} = \frac{d(f,g)}{2^n} \geq \epsilon,$$

i.e., if the Hamming distance $d(f,g)$ between the truth tables of f and g is at least $\epsilon 2^n$. Further, an n-variable Boolean function f will be called ϵ-far from a subset S of n-variable Boolean functions if f is ϵ-far from any function $g \in S$. This definition works perfectly for linearity testing, where S is the set of n-variable linear functions, say.

Unfortunately, this kind of definition, to check whether a function is ϵ-far from the set of n-variable, m-resilient Boolean functions, may not be easy to handle. The main reasons for the limitation in using the distance between the truth tables are as follows.

– The complete characterization of the n-variable m-resilient Boolean functions is not yet known.
– The resiliency definition comes from the Walsh spectrum but not from the truth table.

Thus, for an n-variable function, we may abuse the definition for checking whether it is ϵ-far (not a metric in this abused definition) from the set of m-resilient functions. We define

$$\epsilon = \sum_{s \in S_m} \frac{W_f^2(s)}{2^{2n}}, \text{ where } S_m = \{x \in \{0,1\}^n | wt(x) \leq m\}.$$

One can check that $\epsilon = 0$, if the function is indeed m-resilient. The value of ϵ can be at maximum 1. Looking at Proposition 2 and the discussion after that, it is easy to note that $\epsilon = a^2$ and thus with $O(\frac{1}{\epsilon^{\frac{1}{2}}})$ query complexity, one can decide whether a function is indeed n-variable m-resilient or ϵ-far from such functions.

3 Quantum algorithm for testing Symmetry

The probabilistic classical test for symmetry exploits the condition $f(x) = f(y)$ given $wt(x) = wt(y)$. Thus the probabilistic classical algorithm for testing whether an n-variable Boolean function $f \in \mathcal{SB}_n$ (the set of n-variable symmetric Boolean functions) or not works as in Algorithm 4.

```
Input: A Boolean function $f$ on $n$ variables and the number of iterations $t_w$ for each
       weight $w$, $1 \leq w \leq n-1$
Output: YES/NO
1 For each weight $1 \leq w \leq n-1$ chose an $n$ bit string $a_w$
2 for $i = 1$ to $t_w$ do
3     Randomly choose distinct permutation of $a_w$ say $\pi(a_w)$;
4     Check the condition $f(a_w) = f(\pi(a_w))$;
5     if The condition is not satisfied then
          Report that $f$ is $\epsilon$-far from $\mathcal{SB}_n$ (NO) and terminate;
      end
  end
6 Report that $f \in \mathcal{SB}_n$ (YES);
```
Algorithm 4: Symmetry testing using a classical algorithm

It is well known that if the algorithm reports that f is not symmetric, then it is non-symmetric with probability 1, but if it reports that f is symmetric, then it succeeds with some probability depending on the number of iterations t_w. It is known [14] that if one needs to decide whether a function is ϵ-far from the set of symmetric functions, then the query complexity is $O(\frac{1}{\epsilon})$ to obtain the probability of success greater than or equal to $\frac{2}{3}$ (or any constant c, such that $\frac{1}{2} < c \leq 1$). Recently, it has been shown that [10] in the quantum paradigm, it can be reduced to $O(\frac{1}{\epsilon^{\frac{2}{3}}})$ query complexity. Here we improve the quantum query complexity to $O(\frac{1}{\epsilon^{\frac{1}{2}}})$.

We like to point out that a function $f \in \mathcal{B}_n$ can be ϵ-far from $\mathcal{SB}_n$ for $0 < \epsilon \leq \frac{1}{2}$. This is because, if a function is symmetric then its complement is also symmetric.

3.1 Our proposal

We first take any $n+1$ strings $X_0, X_1, X_2, \ldots, X_n \in \{0,1\}^n$ such that $wt(X_i) = i$. Among them let there be l many points $X_{i_1}, X_{i_2}, \ldots, X_{i_l}$, where the function f evaluates to 1. Let $I_1 = \{i_1, i_2, \ldots, i_l\}$ and $I_0 = [0, \ldots, n] \setminus I_1$. Now, construct an n-variable Boolean function h_0 from f such that,

$$\begin{aligned} h_0(x) &= f(x) \text{ if } wt(x) \in I_0, \\ &= 0 \text{ if } wt(x) \in I_1. \end{aligned} \tag{8}$$

Again we construct another Boolean function h_1 from f such that,

$$\begin{aligned} h_1(x) &= f(x) \text{ if } wt(x) \in I_1, \\ &= 1 \text{ if } wt(x) \in I_0. \end{aligned} \tag{9}$$

Let $\mathbf{0}$, $\mathbf{1}$ be the constant zero and constant one function respectively. It is easy to note that f is symmetric if and only if $h_0 = \mathbf{0}$ and $h_1 = \mathbf{1}$. Let, for $x \in \{0,1\}^n$,

$$\tau_i^0(f) = |\{x : f(x) = 0, wt(x) = i\}|, \ \tau_i^1(f) = |\{x : f(x) = 1, wt(x) = i\}|.$$

It is immediate to see that $\tau_i^0(f) + \tau_i^1(f) = \binom{n}{i}$ and $\sum_{i=0}^n (\tau_i^0(f) + \tau_i^1(f)) = 2^n$. Further, consider that h is the closest symmetric function to f. Then, one may note that $d(f,h) = \sum_{i=0}^n \min\{\tau_i^0(f), \tau_i^1(f)\}$. It is also immediate to see that $d(h_0, \mathbf{0}) = \sum_{i \in I_0} \tau_i^1(f)$ and $d(h_1, \mathbf{1}) = \sum_{i \in I_1} \tau_i^0(f)$. Thus, we have the following technical results.

Proposition 3 *The n-variable function $f \in \mathcal{SB}_n$ if and only if $h_0 = \mathbf{0}$ and $h_1 = \mathbf{1}$. Further, $d(h_0, \mathbf{0}) + d(h_1, \mathbf{1}) \geq d(f, h)$.*

Proof The first statement is easy to follow. Now we prove the second statement. We have, $d(h_0, \mathbf{0}) = \sum_{i \in I_0} \tau_i^1(f)$ and $d(h_1, \mathbf{1}) = \sum_{i \in I_1} \tau_i^0(f)$. Thus, $d(h_0, \mathbf{0}) + d(h_1, \mathbf{1}) = \sum_{i \in I_0} \tau_i^1(f) + \sum_{i \in I_1} \tau_i^0(f) \geq \sum_{i=0}^n \min\{\tau_i^0(f), \tau_i^1(f)\} = d(f, h)$. □

This gives us the following important result for testing the symmetry of a Boolean function.

Theorem 3 *Let f be either symmetric or ϵ-far (for some given $\epsilon > 0$) from $\mathcal{SB}_n$. Given this, the function f is ϵ-far ($\epsilon > 0$) from the set of symmetric Boolean functions, if and only if h_0 is $\frac{\epsilon}{2}$-far from $\mathbf{0}$ or h_1 is $\frac{\epsilon}{2}$-far from $\mathbf{1}$.*

Proof From the second statement of Proposition 3, $d(h_0, \mathbf{0}) + d(h_1, \mathbf{1}) \geq d(f, h)$ and so if f is ϵ-far from h, the closest symmetric function, then h_0 is $\frac{\epsilon}{2}$-far from $\mathbf{0}$ or h_1 is $\frac{\epsilon}{2}$-far from $\mathbf{1}$.

Now we prove the other direction. Let h_0 be $\frac{\epsilon}{2}$-far from $\mathbf{0}$ or h_1 be $\frac{\epsilon}{2}$-far from $\mathbf{1}$. From the first statement of Proposition 3, if h_0 is not $\mathbf{0}$ or h_1 is not $\mathbf{1}$ then f is non symmetric. However it is given that, if f is not symmetric, then it is ϵ-far from the set of symmetric functions. This gives the proof. □

From the definition of h_0 and h_1 it is clear that they are almost the same as symmetric Boolean functions, except for those points x where we have to put the value of $f(x)$ as an output. The symmetric Boolean functions can be efficiently

```
  Input: An $n$-variable Boolean function $f$
  Output: YES/NO
1 Obtain $h_0$ and $h_1$ from $f$ following (8), (9) respectively;
2 Prepare the unitary transformations $U_{h_0}$ and $U_{h_1}$ from $h_0$ and $h_1$;
3 Check in parallel whether $h_0$ and $h_1$ are constant or not, using the constant checking
  algorithm;
4 if at least one of $h_0$ or $h_1$ is not constant then
      Report that the function is $\epsilon$-far from $\mathcal{SB}_n$ (NO) and terminate;
  else
      Report that $f \in \mathcal{SB}_n$ (YES);
  end
```

Algorithm 5: Symmetry testing using a quantum algorithm

implemented, as described in [6]. Thus, we can conclude that the circuits of Boolean functions h_0 and h_1 can be constructed from the circuit of f efficiently.

Hence we note that the algorithm for testing symmetry of f can be efficiently reduced to an algorithm for testing whether h_0 is $\mathbf{0}$ or h_1 is $\mathbf{1}$. This we present formally in Algorithm 5.

Next we need to have an algorithm for testing whether a Boolean function is constant or not.

3.2 Quantum algorithm for testing whether a Boolean function is constant

The famous Deutsch-Jozsa [7] algorithm, in constant number of queries, can identify whether an n-variable Boolean function f is either constant or balanced. Note that this requires the premise that the function will either be balanced or constant. However, the situation here is different as we want to know whether a function is constant or not. An n-variable Boolean function f' is constant if and only if its Walsh spectrum value at the point 0 is $\pm 2^n$. The algorithm for testing whether a given Boolean function f' is constant or δ-far from constant is almost the same as the algorithm for resiliency checking, described in Algorithm 3.

In this case, the function g is such that $g(x_1, x_2, \dots, x_n) = x_1 \vee x_2 \vee \dots \vee x_n$. That is, here we have to take the oracle operator $\mathcal{O}_g$ as

$$\begin{aligned} \mathcal{O}_g|x\rangle &= |x\rangle \text{ if } x = (0,0,\dots,0), \\ &= -|x\rangle \text{ otherwise.} \end{aligned}$$

In this case, the unitary transformation $U_{f'}$ is defined as,

$$\begin{aligned} U_{f'}|x\rangle &= -|x\rangle \text{ if } f'(x) = 1, \\ &= |x\rangle \text{ otherwise.} \end{aligned}$$

Now, the algorithm for testing whether f' is constant or not is given in the following algorithm which is similar to Algorithm 3. Note that a function $f' \in \mathcal{B}_n$ can be δ-far from the constant function for $0 < \delta \leq \frac{1}{2}$. This is because, if a function is constant then its complement is also constant.

The value of t_i's are the same as defined in equation 6. Now let us discuss the query complexity of Algorithm 6. If a function f' is δ-far from a constant function, then $W_{f'}(0,0,\dots,0) \leq 2^n - 2 \cdot 2^n \cdot \delta = 2^n(1-2\delta)$. From Parseval's theorem, we

```
    Input: A Boolean function $f'$ on $n$ variables, available in the form of the unitary
           transformation $U_{f'}$, the number of iterations $r$ and a series of positive integers
           $t_i$, $1 \leq i \leq r$ related to the number of the Grover's iterations
    Output: YES/NO
1   for $i = 1$ to $r$ do
2       Apply the Deutsch-Jozsa algorithm using $U_{f'}$, till the step before measurement to
        obtain $|\Psi\rangle = \frac{W_f(0)}{2^n}|0\rangle + \sum_{s\in\{0,1\}^n\setminus\{0\}} \frac{W_f(s)}{2^n}|s\rangle$;
3       By applying the Grover's iteration $t_i$ many times, obtain
        $|\Psi_{t_i}\rangle = [(2|\Psi\rangle\langle\Psi| - I)\mathcal{O}_g]^{t_i}|\Psi\rangle$;
4       Measure $|\Psi_{t_i}\rangle$ in the computational basis to obtain $n$-bit string $u$;
5       if $u \in \{0,1\}^n \setminus \{(0,0,\ldots,0)\}$ then
            Report that the function is $\delta$-far from constant (NO) and terminate;
        end
    end
6   Report that the function is constant (YES);
```

Algorithm 6: Quantum algorithm for checking whether a Boolean function is constant

have $\sum_{\omega\in\{0,1\}^n} W_{f'}^2(\omega) = 2^{2n}$. Thus, $\sum\limits_{s\in\{0,1\}^n\setminus\{(0,0,\ldots,0)\}} \frac{W_f^2(s)}{2^{2n}} \geq 4\delta - 4\delta^2 \geq 2\delta$, as $\delta \leq \frac{1}{2}$. Now, we can analyse this in a similar way to Theorem 1 and show that the query complexity of Algorithm 6 is $O(\frac{1}{(2\delta)^{\frac{1}{2}}})$. Thus, following Theorem 3, we can argue that the query complexity of Algorithm 5 is $O(\frac{1}{\epsilon^{\frac{1}{2}}})$. This improves the work of [10], that requires $O(\frac{1}{\epsilon^{\frac{2}{3}}})$ query complexity.

Remark 4 One important issue is how the complexity is related to n, the number of input variables to the Boolean function in question. As we have discussed earlier, while testing for whether a function is ϵ-far from the set of n-variable symmetric functions $\mathcal{SB}_n$, we have $0 < \epsilon \leq \frac{1}{2}$. If a function is at a constant Hamming distance β from $\mathcal{SB}_n$, then $\epsilon = \frac{\beta}{2^n}$ and thus the Algorithm 5 will require $O(2^{\frac{n}{2}})$ queries. If $\beta = \frac{2^n}{\zeta(n)}$, then the algorithm will require order of $\sqrt{\zeta(n)}$ queries. That is, if $\zeta(n)$ is polynomial in n, then we have a probabilistic quantum algorithm with polynomially many queries. The algorithm will require constant number of queries for the functions which are far away from the set of symmetric functions, i.e., where β is $O(2^n)$, i.e., when ϵ is constant.

4 Quantum algorithm for testing Linearity

Let l be a linear n-variable Boolean function, i.e., $l(x) = \omega \cdot x$ is available in the form of an oracle. We would like to find ω. For a linear function $l(x) = \omega \cdot x$, $W_l(\omega) = 2^n$ and $W_l(z) = 0$, for $z \neq \omega$. Thus the observed state of n bits will clearly output ω itself (with probability $\frac{W_f^2(\omega)}{2^{2n}} = 1$). That is, the Deutsch-Jozsa algorithm solves this problem in constant time. In the classical model, we need $O(n)$ time to find out the ω. This difference and related results have been pointed out in [2].

The probabilistic classical test for linearity is well known as the BLR test [3] that exploits the condition $l(x \oplus y) = l(x) \oplus l(y)$ for a linear function l, where $a_0 = 0$. However, if $a_0 = 1$ for an affine function ℓ, then we have the condition

$\ell(x \oplus y) = 1 \oplus \ell(x) \oplus \ell(y)$. One may note that one can easily decide whether $a_0 = 0$ or 1 by checking the output of the function at the all-zero, i.e., $(0, 0, \ldots, 0)$ input. Thus the probabilistic classical algorithm for testing whether an n-variable Boolean function f is affine or not works as follows.

```
  Input: A Boolean function $f$ on $n$ variables
  Output: YES/NO
1 $a_0 = f(0, 0, \ldots, 0)$;
2 for $t$ many times do
3   | Randomly choose distinct $x, y \in \{0, 1\}^n$;
4   | Check the condition $f(x \oplus y) = a_0 \oplus f(x) \oplus f(y)$;
5   | If the condition is not satisfied, report that $f$ is not affine (NO) and terminate;
  end
6 Report that the function is affine (YES);
```

Algorithm 7: Classical algorithm for checking whether a Boolean function is affine.

It is well known that if the algorithm reports that f is non-affine, then it is non-affine with probability 1, but if it reports that f is affine, then it succeeds with some probability depending on the number of iterations t. A simple analysis shows that if one needs to decide whether a function is ϵ-far from the set of affine functions, then the probability of success is greater than or equal to $\frac{2}{3}$ (or any constant c, such that $\frac{1}{2} < c < 1$) where t is $O(\frac{1}{\epsilon})$. However, the detailed analysis of this probability of success is quite involved and one may refer to [1, 11] in this direction.

Consider that an n-variable function f is ϵ-far from $\mathcal{A}_n$, the set of all n-variable affine functions. That means, $nl(f) \geq \epsilon 2^n$. Using Parseval's result, it is easy to note that $nl(f) \leq 2^{n-1} - 2^{\frac{n}{2}-1}$. The upper bound can be achieved for functions on even number of variables which are known as bent functions. However, the problem is yet to be settled for the cases on odd number of variables. This tells us that a function on n variables can be ϵ-far from the set of affine functions where $0 \leq \epsilon \leq \frac{1}{2} - \frac{1}{2^{\frac{n}{2}+1}}$. For details of combinatorial, cryptographic and coding theoretic results related to Boolean functions, one may see [12] and the references therein. In general, as $\frac{1}{2^{\frac{n}{2}+1}}$ tends to 0 for large n, we will consider $0 < \epsilon < \frac{1}{2}$ in this case.

4.1 Our proposal

For our purpose, we start with the following technical result.

Theorem 4 *Let $f \in \mathcal{B}_n$ be either affine or ϵ-far (for some given $\epsilon > 0$) from $\mathcal{A}_n$. Let ω be the n-bit pattern measured in step 5 of the Algorithm 1 (the Deutsch-Jozsa algorithm) and let $f'(x) = f(x) \oplus \omega \cdot x$. Given this, the function f is ϵ-far from $\mathcal{A}_n$, if and only if $f'(x)$ is ϵ-far from $\mathcal{C}_n$.*

Proof We first exploit the Deutsch-Jozsa algorithm (refer to step 5 of Algorithm 1) to find a point $\omega \in \{0, 1\}^n$ such that $W_f(\omega)$ is nonzero.

The function f is affine (of the form $\omega_0 \oplus \omega \cdot x$, where $\omega_0 \in \{0,1\}$) if and only if $W_f(\omega) = \pm 2^n$ and for any other $\zeta \in \{0,1\}^n$, such that $\zeta \neq \omega$, $W_f(\zeta) = 0$. Thus f is affine if and only if $f'(x)$ is constant.

Now consider that $f(x) \notin \mathcal{A}_n$ and $f(x)$ is ϵ-far from $\mathcal{A}_n$. Let $\ell(x) = \mu_0 \oplus \mu \cdot x$ be the closest affine function from $f(x)$, where $\mu_0 \in \{0,1\}$ and $\mu \in \{0,1\}^n$. From the definition, we get that $d(f,\ell) \geq \epsilon 2^n$. Without loss of generality, let us take $d(f,\ell) = \epsilon 2^n$ (otherwise, we could have considered $d(f,\ell) = \epsilon_1 2^n$, where $\epsilon_1 \geq \epsilon$ and proceed similarly). For the n-bit pattern ω (measured in step 5 of the Algorithm 1), we have $W_f(\omega) \neq 0$. In terms of the Walsh spectrum values, this implies that $|W_f(\mu)| \geq |W_f(\omega)|$, i.e., $|W_{f \oplus \mu \cdot x}(0)| \geq |W_{f \oplus \omega \cdot x}(0)| = |W_{f'}(0)|$. Thus, if f is ϵ-far from $\mathcal{A}_n$, f' will be ϵ-far from $\mathcal{C}_n$.

In the other direction, if f' is ϵ-far from $\mathcal{C}_n$ for $\epsilon > 0$, then f' is non-linear and thus f is non-linear. Given that f is either affine or ϵ-far from $\mathcal{A}_n$, this proves that it is ϵ-far from $\mathcal{A}_n$. □

Then one may use Algorithm 6 for checking whether $f'(x)$ is constant or ϵ-far from the constant functions $\mathbf{0}$, $\mathbf{1}$. Thus, given that a Boolean function f is either affine or it is ϵ-far from the set of affine functions, the query complexity of our algorithm is $O(\frac{1}{\epsilon^{\frac{1}{2}}})$. If the algorithm outputs that the function is not affine, then it is certainly true. If it outputs that the function is affine, then it is true with a constant probability. Thus, we improve the work of [10], that requires $O(\frac{1}{\epsilon^{\frac{2}{3}}})$ query complexity.

Remark 5 We need to highlight a subtle difference between the testing the symmetry and linearity. We reduce both the cases in testing whether a function is constant. In case of symmetry, the reduction of a symmetric function to a constant function (as in Theorem 3 does not require any quantum framework and it is done in classical domain only. However, in case of linearity, we require the application of the Deutsch-Jozsa algorithm in Theorem 4 to find a point ω where the Walsh spectrum is nonzero. This can be done in constant number of queries deterministically using the quantum algorithm, which would not have been possible using any classical algorithm.

Similar to Remark 4, we can understand how the complexity is related to n. While testing for whether a function is ϵ-far from the set of n-variable affine functions $\mathcal{A}_n$, we have $0 \leq \epsilon \leq \frac{1}{2} - \frac{1}{2^{\frac{n}{2}+1}}$. If a function is at a constant Hamming distance β from $\mathcal{A}_n$, then $\epsilon = \frac{\beta}{2^n}$ and thus the algorithm will require $O(2^{\frac{n}{2}})$ many queries. If $\beta = \frac{2^n}{\zeta(n)}$, then the algorithm will require $O(\sqrt{\zeta(n)})$ queries. That is, if $\zeta(n)$ is polynomial in n, then we need polynomially many queries. The algorithm will require constant number of queries for highly non-linear functions where β is $O(2^n)$, i.e., when ϵ is constant.

5 Conclusion

In this paper, we study the problem of checking resiliency of a Boolean function in the quantum paradigm. The input to the algorithm is an n-variable Boolean function and the algorithm should output YES if the function is m-resilient and NO if it is not. Our algorithm provides the NO answer correctly, while the YES answer is provided with probability greater than some predefined constant. We use the well known Deutsch-Jozsa and Grover's algorithms for the purpose. Algorithm 3 shows that it requires exponentially many queries but polynomially many measurements in n in the worst case. We also identify a sub-class of Boolean functions for which we require polynomially many queries as well as polynomially many measurements in the worst case. For such a class no efficient classical algorithm is known.

Next we concentrate on symmetric and affine Boolean functions. We show that the problem of checking whether a function is symmetric (respectively affine) can be efficiently reduced to the problem of checking whether a function is constant. Using a similar direction as in resiliency checking, we provide a quantum algorithm in this direction, that improves the recent results explained in [10]. We also interpret the results in property testing framework.

It is interesting to explore how this kind of technique using the Walsh spectrum of Boolean functions, associated with the Deutsch-Jozsa and the Grover's Algorithms, can be exploited for testing some other properties of Boolean functions.

References

1. M. Bellare, D. Coppersmith, J. Hastad, M. Kiwi and M. Sudan. Linearity testing over characteristic two. IEEE Trans. Inform. Theory, 42, 1781 (1996).
2. E. Bernstein and U. Vazirani. Quantum complexity theory. Proceedings of the 25th Annual ACM Symposium on Theory of Computing, (ACM Press, New York, 1993), pp. 11–20.
3. M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. J. Comput. Syst. Sci. 47(3), 549 (1993).
4. M. Boyer, G. Brassard, P. Hoeyer and A. Tapp. Tight bounds on quantum searching. Fortsch. Phys. 46:493-506, 1998 (arXiv:quant-ph/9605034).
5. S. L. Braunstein, B.-S. Choi, S. Ghosh and S. Maitra. Exact quantum algorithm to distinguish Boolean functions of different weights. Journal of Physics A: Mathematical and Theoretical, Volume: 40, Pages 8441-8454, doi:10.1088/1751-8113/40/29/017, published: 3 July 2007.
6. E. Demenkov, A. Kojevnikov, A. Kulikov and G. Yaroslavtsev. New upper bounds on the Boolean circuit complexity of symmetric functions. Information Processing Letters, 110:264–267, 2010.
7. D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. Proceedings of Royal Society of London, A439:553–558 (1992).
8. L. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th Annual Symposium on the Theory of Computing (STOC)*, May 1996, pages 212–219. Available at `http://xxx.lanl.gov/abs/quant-ph/9605043`.
9. X. Guo-Zhen and J. Massey. A spectral characterization of correlation immune combining functions. *IEEE Transactions on Information Theory*, 34(3):569–571, May 1988.
10. M. Hillery and E. Andersson. Quantum tests for the linearity and permutation invariance of Boolean functions. Physical Review A 84, 062329 (2011).
11. T. Kaufman, S. Litsyn and N. Xie. Breaking the ϵ-soundness bound of the linearity test over $GF(2)$. Siam J. Comput., 39(5), 1988 (2010).
12. S. Kavut, S. Maitra and M. D. Yucel. Search for Boolean Functions with Excellent Profiles in the Rotation Symmetric Class. IEEE Trans. Inform. Theory, 53(5), 1743 (2007).
13. S. Maitra and P. Mukhopadhyay. Deutsch-Jozsa Algorithm Revisited in the Domain of Cryptographically Significant Boolean Functions. In International Journal on Quantum Information, Pages 359–370, Volume 3, Number 2, June 2005.
14. K. Majewski and N. Pippenger. Attribute estimation and testing quasi-symmetry. Information Processing Letters 109:233-237, 2009
15. M. A. Nielsen and I. L. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, 2010.
16. P. Sarkar and S. Maitra. Nonlinearity bounds and constructions of resilient Boolean functions. In *Advances in Cryptology - CRYPTO 2000*, number 1880 in Lecture Notes in Computer Science, pages 515–532. Springer Verlag, 2000.

17. Y. V. Tarannikov. On resilient Boolean functions with maximum possible nonlinearity. In *Progress in Cryptology - INDOCRYPT 2000*, number 1977 in Lecture Notes in Computer Science, pages 19–30. Springer Verlag, 2000.
18. Y. Zheng and X. M. Zhang. Improved upper bound on the nonlinearity of high order correlation immune functions. In *Selected Areas in Cryptography - SAC 2000*, number 2012 in Lecture Notes in Computer Science, pages 264–274. Springer Verlag, 2000.